Continuous residual reinforcement learning for traffic signal control optimization

Aslani, Mohammad; Seipel, Stefan; Wiering, Marco

Published in:
Canadian Journal of Civil Engineering

DOI:
10.1139/cjce-2017-0408

Publication date:
2018

Continuous residual reinforcement learning for traffic signal control optimization

Mohammad Aslani*

Department of Industrial Development, IT and Land Management, University of Gävle, Gävle, Sweden;

moh.aslani@gmail.com

Stefan Seipel

Department of Industrial Development, IT and Land Management, University of Gävle, Gävle, Sweden;

Division of Visual Information and Interaction, Department of Information Technology, Uppsala University, Uppsala, Sweden

stefan.seipel@hig.se

Marco Wiering

Institute of Artificial Intelligence and Cognitive Engineering, University of Groningen, Groningen, the Netherlands

m.a.wiering@rug.nl

* Corresponding author. Tel: (+46) 737073770,

Department of Industrial Development, IT and Land Management, University of Gävle, Kungsbäcksvägen 47, 801 76 Gävle, Sweden

Word count: 7800

Abstract

Traffic signal control can be naturally regarded as a reinforcement learning problem. Unfortunately, it is one of the most difficult classes of reinforcement learning problems owing to its large state space. A straightforward approach to address this challenge is to control traffic signals based on continuous reinforcement learning. Although they have been successful in traffic signal control, they may become unstable and fail to converge to near-optimal solutions. We develop adaptive traffic signal controllers based on continuous residual reinforcement learning (CRL-TSC) that is more stable. The effect of three feature functions is empirically investigated in a microscopic traffic simulation. Furthermore, the effects of departing streets, more actions, and the use of the spatial distribution of the vehicles on the performance of CRL-TSCs are assessed. The results show that the best setup of the CRL-TSC leads to saving average travel time by 15% in comparison to an optimized fixed-time controller.

Keywords: Continuous state reinforcement learning, Adaptive traffic signal control, Microscopic traffic simulation.

1-Introduction

The high trend of population growth in cities and consequently a high level of accumulation and concentration of economic and social activities in urban areas lead to a growing demand for transportation (Bhatta 2010, Rodrigue et al. 2017). Such an increase in transportation demand renders the current transportation infrastructures incapable of successfully handling transportation needs.

Heavy traffic congestion and long vehicle queues on signalized approaches are common phenomena which are currently observable every day in large cities. Traffic congestion usually arises from the excess of demand in comparison to the available capacity of the streets. One of the effective solutions for alleviating traffic congestion is to embed intelligent transportation system (ITS) technologies into the traffic infrastructures in order to make them work more efficiently (Chowdhury and Sadek 2003). One of the cornerstones of ITS that attracted attention of a lot of researchers and practitioners is developing adaptive traffic signals.

Adaptive traffic signal control is a real-time traffic management strategy in which traffic signal timing changes, or adapts, according to the actual traffic demand. It uses the observed information to immediately adapt to traffic demand (Aslani et al. 2017, El-Tantawy et al. 2013). In this context, multi-agent systems (MAS) have become very popular in traffic control owing to the analogies of their characteristics (e.g. distribution, intelligence, and autonomy) with the traffic control nature (Oliveira and Camponogara 2010). In this study, which uses MAS to control traffic, there are two types of agents: traffic signal agents (active agents) which have learning capabilities and vehicle agents (passive agents) which have behaviors of acceleration, deceleration, and overtaking.

Due to the complexity and uncertainty arising in traffic environments, it is difficult to resolve the problem with preprogrammed multi-agent behaviors. Therefore, a learning mechanism is required by which the agent can gain the necessary knowledge while making a decision and interacting intelligently with an uncertain environment. Within such a context, reinforcement learning serves as a promising approach for training agents such that agents never see examples of the correct behavior, but instead receive positive or negative rewards for the actions they try (Kaelbling et al. 1996, Sutton and Barto 1998). It allows agents to automatically determine the ideal behavior in order to maximize their performance (Schwartz 2014, van Otterlo and Wiering 2012). Numerous reinforcement learning algorithms have been developed in the literature; however, the temporal difference learning methods (Sutton 1988) are the most relevant to the traffic signal control problem.

Conventional reinforcement learning methods need to first discretize the state space and then apply a suitable algorithm for a discrete stochastic system. This discretization has some drawbacks. A coarse discretization results in a jerky output and poor performance, while a fine discretization, which may lead to better performance, necessitates not only a large memory storage but also many learning trials. In order to eliminate these difficulties, we develop adaptive traffic signal controllers founded on continuous reinforcement learning. Continuous reinforcement learning rests on the concept of generalization through function approximators (Sutton and Barto 1998). The idea behind it is that the agent requires no direct experiencing of all states since the values of state-actions are estimated from the values of similar state-actions (Szepesvári 2010). In continuous reinforcement learning, the original state space is mapped onto a feature space through a feature function. The performance of the continuous reinforcement learning methods is highly dependent on the suitability of the selected feature function. With this

end in mind, three different feature functions, namely tile coding, triangular-shaped functions (TSFs), and radial basis functions (RBFs) are compared. The combination of reinforcement learning with function approximation may become unstable and fail to converge to a near-optimal solution. To overcome this challenge, the design of adaptive traffic signal controllers is done through residual algorithms (Baird 1999).

Outline of the paper. The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 describes the principles of reinforcement learning. The proposed adaptive traffic signal controller based on continuous residual reinforcement learning is presented in Section 4. Section 5 presents the experimental setup and results. Section 7 provides a discussion of our findings and Section 8 concludes the paper.

2-Related work

Adaptive traffic signal control optimizes the traffic signal scheduling parameters based on current traffic conditions to achieve a set of specific goals. Different methods have been proposed in traffic engineering to adaptively control traffic signals, e.g. SCOOT (Hunt et al. 1981), SCAT (Sims and Dobinson 1980), OPAC (Gartner 1983), PRODYNE (Henry et al. 1983), and RHODES (Head et al. 1992). In recent decades, different methods in artificial intelligence have attracted the interest of experts in traffic signal control. Neural networks (Bishop 1995, Samarasinghe 2016), fuzzy inference systems (Mamdani 1974, Takagi and Sugeno 1985, Zadeh 1965), and reinforcement learning (Sutton and Barto 1998) are three machine learning approaches drawn upon for developing adaptive traffic signal controllers.

In the research done by Srinivasan et al. (2006), two traffic signal control systems based on neural networks were developed. The first system was developed using the integration of simultaneous perturbation stochastic approximation (SPSA) and a fuzzy neural network. In this

method, SPSA was used in modeling the weight update process of a five-layer fuzzy neural network. The hybrid neural network based on a multi-agent system, as the second system, was developed to solve the online distributed control problem. Each agent includes a five-layer fuzzy neural network for online decision-making. The learning process contains three steps, namely reinforcement learning, adjustment of learning rates and weights, and adjustments of fuzzy relations. A microscopic traffic simulation of the central business district of Singapore was developed to be used as a test-bed for assessing the performance of the systems. The results demonstrated that the second system outperforms the first one in more complex scenarios with multiple traffic peaks.

Chiou and Huang (2013) employed the integration of a fuzzy inference system and a stepwise genetic algorithm to develop adaptive traffic signal controllers. Since fuzzy inference systems are not able to learn as such and requires that the knowledge base is derived from experts' knowledge, the stepwise genetic algorithm was deployed to tune both the form of fuzzy membership functions and fuzzy rules. Also, traffic flows and queue lengths were selected as the input variables and the extension of green time was chosen as the control variable. The control system was tested in an isolated intersection and a 1×3 traffic network. Through the experimental results, they conclude that the proposed system is efficient and robust.

In (Bi et al. 2014), a distributed traffic signal control system founded on type-2 fuzzy logic control was adopted. A differential evolution method was deployed to tune the knowledge base. The proposed method was benchmarked against type-1 fuzzy logic control and fixed-time methods on a grid-type network composed of eleven intersections. The results revealed that the proposed method has better performance.

In this research, reinforcement learning is employed to develop adaptive traffic signal controllers. Within such a context, Wiering (2000) employed modeled-based reinforcement learning to minimize the waiting time. Vehicles have the ability to communicate with traffic signals. The average waiting time estimated by vehicles is transmitted to the traffic signal located at the next intersection. Then, a traffic signal selects a phase with the maximum sum of the vehicles' gains. The gain is defined as the difference between the vehicle's waiting time when the light is red and when it is green. Results indicated that the proposed method reduces waiting time by 22% in comparison to a fixed time controller.

In (Steingrover et al. 2005), the authors extended Wiering's approach using extra information from neighboring intersections. Although adding new information of the congestion on the next lanes allows the agents to learn how to handle traffic when the departing links are congested, it makes the state-space bigger and decreases the convergence speed.

Salkham et al. (2008) proposed a traffic signal control system using collaborative reinforcement learning in which each signalized intersection learns the suitable phase timing by collaborating with neighboring controllers. A pair of the phase number and its status (busy/not busy) was considered as the state space. Also, the action of each controller was the phase duration. The performance of the proposed method was evaluated in a real-world simulated traffic environment of downtown Dublin. It was benchmarked against a fixed-time system and a SAT-like algorithm (Richter 2006) that emulates SCATS' behavior of saturation balancing. The experiments showed that the proposed system significantly outperforms other methods in terms of average waiting time.

Medina et al. (2010) used reinforcement learning to adaptively control traffic signals. Each traffic signal controller senses the number of vehicles approaching its intersection. The state

space is augmented by the numbers of vehicles stopped on departing lanes approaching adjacent intersections. The results revealed that the proposed adaptive traffic signal controllers lead to lower delays as well as a more balanced distribution of the delay among all vehicles in comparison to a fixed-time method.

Medina and Benekohal (2012) employed the Q-learning algorithm and an approximate dynamic programming algorithm to develop traffic signal control strategies. At each intersection, the learning controller takes into account not only the local state but also the congestion state of neighboring intersections. A real-world traffic simulation was carried out in VISSIM to test the efficiency of the proposed systems. The proposed systems were compared with TRANSYT7F and the results indicated that they lead to 13% lower average delay.

In (El-Tantawy et al. 2013), a coordinated traffic signal control scheme based on multi-agent reinforcement learning was developed. In this scheme, each agent that controls one intersection coordinates its actions with neighboring intersections. The state space contains the index of the current green phase, elapsed time of the current phase, and maximum queue lengths associated with each phase. The action of each agent is to extend the current phase or to switch to another phase. The performance of the proposed scheme was evaluated in a simulated network of 59 intersections in downtown Toronto. The results revealed that their method outperforms the current control scheme of the study area by 26% regarding average travel time.

Employing discrete state reinforcement learning for traffic signal control which is naturally continuous may bring about a low convergence speed and poor performance. The more reasonable solution is to employ continuous reinforcement learning that has the ability to perform accurately on unseen data. Within such a context, Prashanth and Bhatnagar (2011) enabled the traffic signal controller to handle large state space by the combination of Q-learning

and a function approximator. Queue lengths and elapsed time for the red signal are variables forming the state space. The objective of the controller is to minimize the queue lengths by considering fairness among different approaching links such that no lane has a long red time. The results showed that Q-learning with function approximation significantly outperforms the fixed time controller.

In (Abdoos et al. 2014), the authors proposed a hierarchical multi-agent reinforcement learning architecture to provide different levels of control for a traffic network. The intelligent agents are divided into two groups: local agents that are responsible for controlling each intersection and global agents that adjust actions of the local agents. Local agents and global agents adapt to prevailing traffic conditions through standard Q-learning and continuous Q-learning respectively. There are nine local agents (3×3 junction grid) and three global agents that each supervises three local agents. The results revealed that the proposed method outperforms standard Q-learning in terms of delay time.

In this research, we develop adaptive traffic signal controllers based on continuous residual reinforcement learning (CRL-TSC) that is more stable and the performance of the best CRL-TSC is compared with fixed-time, standard Q-learning, and actor-critic controllers. Also, the effect of three feature functions, namely tile coding, radial basis functions (RBFs), and triangular-shaped functions (TSFs) are empirically investigated. Moreover, the impacts of considering departing links and the spatial distribution of vehicles in the state space, as well as more actions in the action space are evaluated. Departing link variables provide CRL-TSCs with the ability to handle the spillback phenomenon and indirect cooperation with neighboring intersections. The spatial distribution causes the distance of the vehicles to the associated intersection to be, to some

extent, regarded. Investigating the effect of more actions determines if increasing the flexibility in the action space improves the performance.

3-Reinforcement learning

Reinforcement learning originally stems from the study of animal intelligence (Thorndike 1911) and has been developed as a major branch of machine learning for solving sequential decision-making problems. Reinforcement learning is an approach to learn an optimal policy of an agent by interacting with its surrounding environment such that it maximizes some numerical value that represents a long-term objective.

In reinforcement learning, the decision maker is called an intelligent autonomous agent and everything except the agent is referred to as the environment (Sutton and Barto 1998). In many cases, the environment has the Markovian property with respect to the agent's perception. The Markovian property means that the result of an action does not depend on all previous actions and visited states (history), but only depends on the current state. A fundamental formalism for reinforcement learning, especially in stochastic domains is called a Markov Decision Process (MDP) (van Otterlo and Wiering 2012). In fact, the basic elements of the reinforcement learning problem can be formalized by an MDP. MDP consists of four elements: S, A, $R^a_{ss'}$, and $P^a_{ss'}$, where S is the set of states, A is the set of agent`s actions, $P^a_{ss'}$ is the probability of going from state s to s' after taking action a, and $R^a_{ss'}$ is the average reward for the transition from state s to s' by taking action a. The decision-making function of the agent that specifies which action should be taken in each state is called the policy π(s, a). In other words, the policy is a mapping from states to actions and indicates the probability of selecting action *a* in state s. In this research, we employ Boltzmann policy in order to balance between exploration and exploitation.

Another element of reinforcement learning is the use of action values. While a reward function shows how good an action is in an immediate term, an action value specifies how good a particular action is in a long-term sense. The action value that shows the value of performing an action in a state and thereafter following the policy π is calculated by equation 1.

$$Q^{\pi}(s,a)=E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \vee s_t=s, a_t=a\right\} \quad (1)$$

Where $Q^{\pi}(s,a)$ is the state-action value, that corresponds to the expected return when starting in the state *s* and taking action *a* and following the policy π thereafter. r_{t+k+1} is the reward obtained when arriving into states s_{t+1}, s_{t+2} etc. γ is the discount factor that represents the difference in importance between future rewards and instant rewards. The objective of reinforcement learning is to find a policy which maximizes the action values. The state space of the traffic environment is very large and continuous, and this makes conventional reinforcement learning inefficient. In this research, continuous reinforcement learning is employed in order to tackle this challenge.

4-Continuous residual reinforcement learning traffic signal controller

The continuous residual reinforcement learning traffic signal controller (CRL-TSC) is an autonomous learner that iteratively interacts with the traffic environment in order to find the optimal or near-optimal signal timing plan. CRL-TSC tunes the parameters of the traffic signal controller in response to the changing traffic conditions. At the beginning of each phase, each CRL-TSC senses the current traffic state (s_t) of its local intersection. The traffic state is represented by a vector whose elements are the number of vehicles on each approaching street. Through this representation, the traffic load which is easy to be measured through existing

sensors is encoded in the definition of the environment state. Also, this state definition allows us to manage vehicles with many passengers (e.g. buses). It should be noted that each dimension of the state space is normalized between 0 and 1.

After sensing the current traffic state, CRL-TSC selects a green time duration (action), i.e., a value from [20, 30, 40, 50, 60, 70, 80, 90] seconds (a_t). Once a green time is selected, CRL-TSC waits to the end of the current phase duration which is the summation of the green and yellow interval. Then, CRL-TSC receives a reward signal (r_{t+1}). The reward signal provided to CRL-TSC is defined as the negative total number of the vehicles waiting on all the streets leading to the associated junction. Using this reward function causes assigning longer green time to the streets with heavier traffic congestion and higher input traffic flows. In fact, if the selected green time leads to passing (eliminating) more vehicles from the streets with a high input traffic flow, it receives a greater reward. To put it simply, it considers both traffic congestions and the input traffic flow of the approaching streets.

Once the immediate reward signal (r_{t+1}) is received, CRL-TSC senses the new traffic condition (s_{t+1}) and selects another green time duration (a_{t+1}) for the next traffic light configuration.

Through r_{t+1}, s_{t+1}, and a_{t+1} the value of (s_t, a_t) is estimated. The generalization concept is drawn upon for estimating the value of each state-action pair. It enables traffic signals to perform successfully on unseen states. In fact, there is a natural metric on the state space in such a way that close states require similar behaviors. Thus, CRL-TSCs are able to contend with states never exactly experienced before and they can learn efficiently by generalizing from previously (similar, close) experienced states. Indeed, CRL-TSCs require no direct experience of all different environment states and can obtain the value of a state from that of other similar states.

The value of each state-action pair to be approximated at time t under policy π, $Q^{\pi}(s_t, a_t)$, is represented as a linear function $\widehat{Q}^{\pi}(s_t, a_t) = \theta^T \phi(s_t, a_t)$ where s_t is the state at time t, θ is a set of scalar weights and ϕ is a feature function which encodes the similarity relationship of the state-actions with that of their values (Sutton and Barto 1998). Both θ and ϕ are (n×k)-dimensional vectors where n is the total number of features and k is the number of actions.

$\phi(s_t, a_t)$ is defined according to equation 2.

43 $$\phi^T(s_t, a_t) = [\varphi_1(s_t).b_1, \ldots, \varphi_n(s_t).b_1, \varphi_1(s_t).b_2, \ldots, \varphi_n(s_t).b_2, \ldots, \varphi_1(s_t).b_k, \ldots, \varphi_n(s_t).b_k]$$

$$b_i = \begin{cases} 1, if\ a_t = a_i \\ 0, if\ a_t \neq a_i \end{cases} \quad (2)$$

Choosing the right feature function type is critical for successful learning. Among different feature functions employed in linear function approximators, tile coding, RBF, and TSF are the most exploited techniques. Tile coding generalizes the state space into partitions called tilings (Albus 1975). Each tiling consists of a set of non-overlapping grid cells called a tile. The membership value of the triggering state to different tiles is either 0 or 1 (equation 3). There is always just one feature active in each tiling layer. Let N is the dimension of the state space and m_j is the number of tiles on j-th dimension. The tile coding features are created as follows:

$$\varphi_i(s_t) = \begin{cases} 0\ if\ s_t \notin tile_i \\ 1\ if\ s_t \in tile_i \end{cases} 1 < i < n, n = m_1 \times m_2 \times m_3 \ldots \times m_N \quad (3)$$

Their reliance on a set of binary features makes tile coding one of the most explored feature functions. In this research, each state variable is partitioned into a finite set of tiles and then the tiling is created by combining the tiles in each state variable in a vector. Each tiling has the same number of tiles in each dimension.

RBF provides a continuous representation of states instead of a binary representation. In fact, RBF builds more a complex representation using a distance metric leading to non-binary features. The activation of each RBF feature continuously decays away from the center of the RBF. The output of the i-th RBF centered around s_t is calculated according to equation 4.

$$\varphi_i(s_t) = e^{-\sum_{j=1}^{N} \frac{(s_t^j - \mu_{ij})^2}{2\sigma_{ij}^2}} \quad 1 < i < n, n = m_1 \times m_2 \times m_3 \ldots \times m_N \tag{4}$$

Where σ_{ij} and μ_{ij} are the standard deviation and center of RBF_i on the j-th dimension and s_t^j is the j-th dimension of the state at time t. Figure 1 shows the parameters of RBFs and how RBFs are located on each dimension. As it is clear, the distance between the centers of two consecutive RBFs and the standard deviation of each RBF on the j-th dimension are $\frac{1}{m_j}$ and $\frac{1}{2m_j}$ respectively where m_j is the number of RBFs on the j-th dimension. In fact, the centers of RBFs are distributed in the state space as a fixed uniform grid.

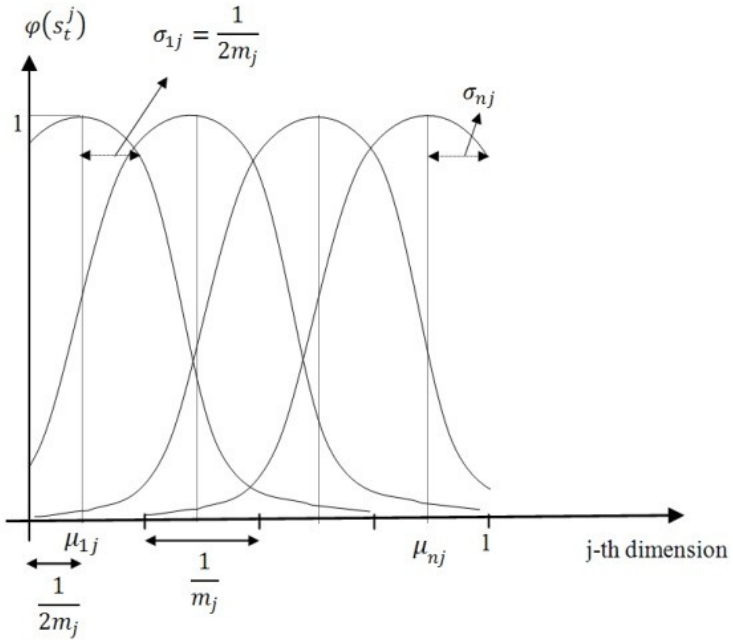

Figure 1. Layout of RBFs and their parameters on the j-th dimension

TSF is a function whose figure takes the shape of a triangle. Equation 5 is used to calculate the degree of membership to different TSFs in each state variable.

$$\varphi_i(s_t)=\prod_{j=1}^{N}\left(1-\left|s_t^j-\mu_{ij}\right|.m_j\right)\ 1<i<n, n=m_1\times m_2\times m_3\ldots\times m_N \qquad (5)$$

The output of a TSF is zero when state s is far from the center (μ). Figure 2 indicates the arrangement of TSFs on the j-th dimension. It is evident that maximally two features per dimension become active in each state.

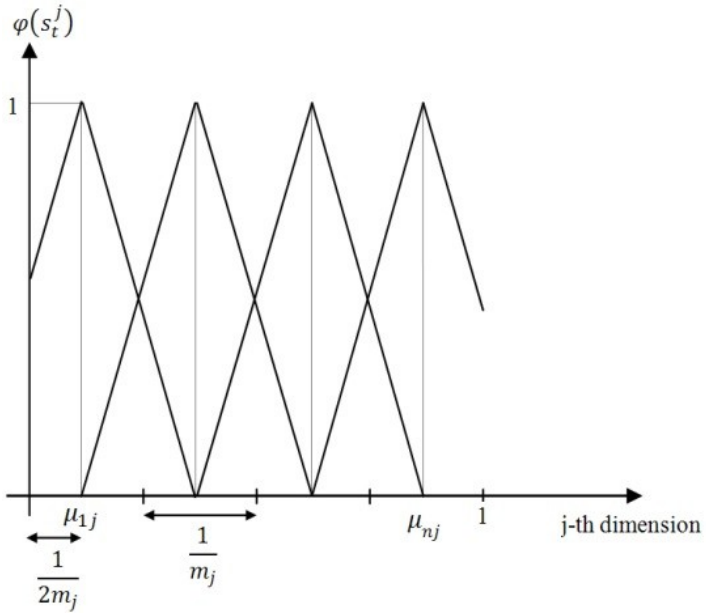

Figure 2. Layout of TSFs and their parameters on the j-th dimension

The number of tiles, RBFs, and TSFs, as well as the locations of RBFs and TSFs, centers greatly affect the accuracy and validity of the learning performance of CRL-TSCs. Also, the poorly placed tiles, RBFs, and TSFs can prevent CRL-TSCs to correctly estimate the value function even in some simple domains. Therefore, in our experiments, we generate different feature functions of tile coding, RBF, and TSF with a different density of features on each state variable.

In fact, we evaluate and compare the performance of CRL-TSCs by considering different numbers of tiles, RBFs, and TSFs on each dimension (see Section 5-1).

In all feature functions, after the values of features are calculated, they are normalized so that the total sum of them becomes 1. The scalar weights vector *(θ)* is updated such that the following cost function is minimized (equation 6).

$$C = E\left[\left(r_{t+1} + \gamma\,\theta^T.\phi\left(s_{t+1}, a_{t+1}\right) - \theta^T.\phi\left(s_t, a_t\right)\right)^2\right] \qquad (6)$$

A good strategy for minimizing equation 6 is to try to minimize it on the observed examples. Stochastic gradient descent is able to do this by tuning the scalar weight vector after each example observed. Therefore, by using stochastic gradient descent, θ is updated according to equation 7.

$$\Delta\theta = \alpha\left(r_{t+1} + \gamma\,\theta_t^T.\phi\left(s_{t+1}, a_{t+1}\right) - \theta_t^T.\phi\left(s_t, a_t\right)\right).\nabla_\theta Q\left(s_t, a_t\right)$$

$$\theta_{t+1} = \theta_t + \Delta\theta$$

(7)

In this equation, α is the learning rate and $\nabla_\theta Q\left(s_t, a_t\right) = \phi\left(s_t, a_t\right)$. Although this method is a very simple and fast way for updating θ, it is not guaranteed to converge due to the fact that changing the value of one state usually changes the values of other states such as that of the state s_{t+1}. Consequently, the estimated value ($\hat{Q}^\pi\left(s_t, a_t\right)$) may move away from its target value. In order to tackle this problem, the scalar weight vector (θ) can also be updated according to equation 8.

$$\Delta\theta' = \alpha\left(r_{t+1} + \gamma\,\theta_t^T.\phi\left(s_{t+1}, a_{t+1}\right) - \theta_t^T.\phi\left(s_t, a_t\right)\right).\left(\nabla_\theta Q\left(s_t, a_t\right) - \gamma\nabla_\theta Q\left(s_{t+1}, a_{t+1}\right)\right)$$

$$\theta_{t+1} = \theta_t + \Delta\theta'$$

(8)

Where $\nabla_\theta Q(s_{t+1}, a_{t+1}) = \phi(s_{t+1}, a_{t+1})$. This residual learning method considers the states s_t and s_{t+1} in order to improve stability and convergence properties. However, this method does not always learn as quickly as equation 7. In fact, equation 7 is fast but unstable; equation 8 can be stable but slow in terms of convergence. Therefore, the best solution is to combine the two methods in order to gain the advantages of them (fast and stable learning). This can be achieved by using a weighted average of two gradient vectors (equation 9).

$$\Delta\theta'' = (1-\beta)\Delta\theta + \beta\Delta\theta'$$

$$\Delta\theta'' = \alpha\left(r_{t+1} + \gamma\,\theta_t^T.\phi(s_{t+1}, a_{t+1}) - \theta_t^T.\phi(s_t, a_t)\right).\left(\nabla_\theta Q(s_t, a_t) - \beta\gamma\nabla_\theta Q(s_{t+1}, a_{t+1})\right)$$

$$\theta_{t+1} = \theta_t + \Delta\theta''$$

(9)

Where $\beta \in [0,1]$ attenuates the effect of the successor state (s_{t+1}). In this research, β is adapted during the learning process using equation 10 (Baird 1999).

$$\beta = \frac{\Delta\theta.\Delta\theta'}{\Delta\theta.\Delta\theta' - \Delta\theta'.\Delta\theta'}$$

(10)

Another point to be noted is the way how CRL-TSCs select the suitable actions in each traffic state. Basically, action selection should be based on the value of state-action pairs ($Q^\pi(s_t, a_t)$). However, owing to the fact that CRL-TSCs do not possess the correct values of each state-action pair at the beginning of the learning process, they need to explore different green time durations regardless of their values in order to achieve accurate estimations of state-action values. As time (t) goes by and CRL-TSCs obtain better estimations, they should rely less on an exploration through random green time selection and begin to exploit their obtained knowledge of the traffic environment by choosing those green times which possess a fairly high value. In

this research, in order to trade-off between exploration and exploitation the Boltzmann exploration strategy is employed. The probability of selecting each available action is calculated according to equation 11.

$$Pr(s_t, a_i) = \frac{\exp(\omega . Q(s_t, a_i))}{\sum_{j=1}^{k} \exp(\omega . Q(s_t, a_j))}$$

(11)

Where k is the number of actions, a_i is each action available in state s_t, and the parameter ω controls the exploration rate. The higher the ω value, the sharper the distribution becomes. For $\omega \to \infty$, it converges to the greedy policy. By using the Boltzmann policy, actions with high values are more likely to be selected than actions with a lower value. Algorithm 1 describes how each CRL-TSC works.

Algorithm 1. CRL-TSC

Initialize θ, ϕ, α, γ, and ω
A is the action set
t ← 0
loop
    s_t, a_t ← initial state and action of the episode
    repeat
        Set the current phase duration to a_t+¿time
        Wait until the end of the phase
        Observe the number of vehicles on each approaching street
        Calculate reward r_{t+1}
        $\delta_{t+1} \leftarrow r_{t+1} - \theta^T \phi(s_t, a_t)$
        Estimate the state s_{t+1}

for all $a_i \in A$ do

$Q(s_{t+1}, a_i) \leftarrow \theta^T \phi(s_{t+1}, a_i)$

$\rho \leftarrow \rho + \exp(\omega . Q(s_{t+1}, a_i))$

end for

Uniformly draw a number $P \in [0, 1]$

$d \leftarrow 0$

for all $a_i \in A$ do //Boltzmann

$Pr(s_{t+1}, a_i) \leftarrow \frac{\exp(\omega . Q(s_{t+1}, a_i))}{\rho}$

if $P \leq Pr(s_{t+1}, a_i) + d$

$a_{t+1} \leftarrow a_i$

break

else

$d \leftarrow d + pr(s_{t+1}, a_i)$

end if

end for

$Q(s_{t+1}, a_{t+1}) \leftarrow \theta^T \phi(s_{t+1}, a_{t+1})$

$\delta_{t+1} \leftarrow \delta_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$

$\Delta\theta \leftarrow \alpha \delta_{t+1} \phi(s_t, a_t)$,

$\Delta\theta' \leftarrow \alpha \delta_{t+1} (\phi(s_t, a_t) - \gamma \phi(s_{t+1}, a_{t+1}))$

$\beta \leftarrow \frac{\Delta\theta . \Delta\theta'}{\Delta\theta . \Delta\theta' - \Delta\theta' . \Delta\theta'}$

$\Delta\theta'' \leftarrow (1-\beta)\Delta\theta + \beta \Delta\theta'$

$\theta_{t+1} \leftarrow \theta_t + \Delta\theta''$

$t \leftarrow t+1$

until s_t in terminal

end loop

5-Simulation experiments

5-1- Implementation

The microscopic traffic simulation in this research comprises a traffic network, vehicles, and CRT-TSCs. The employed 3×3 grid network for which one CRL-TSC controls one intersection is depicted in figure 3. All the streets are two-way (bi-directional) with two lanes at each side. The capacity of each lane is 40 vehicles. The length of each street is 250 meters. Vehicles enter the network using a Gaussian distribution through 12 sources that lie on the borders of the network. In each intersection, it is assumed that among all the vehicles approaching the intersection, 33.3% of them go straight, 33.3% turn left, and 33.3% turn right. The traffic network configuration parameters are shown in table 1. We have used this small grid in order to accomplish a careful analysis of the impact of different parameters on the performance. However, the proposed CRL-TSC can be easily used in larger traffic networks.

Table 1. Traffic network configuration

Properties	Value
number of intersections	9
number of links	48
average length of links	250
number of lanes per links	2
maximum speed	50 km/h
number of input/output centroids	12
arrival distribution	Gaussian
simulation duration	700 hours

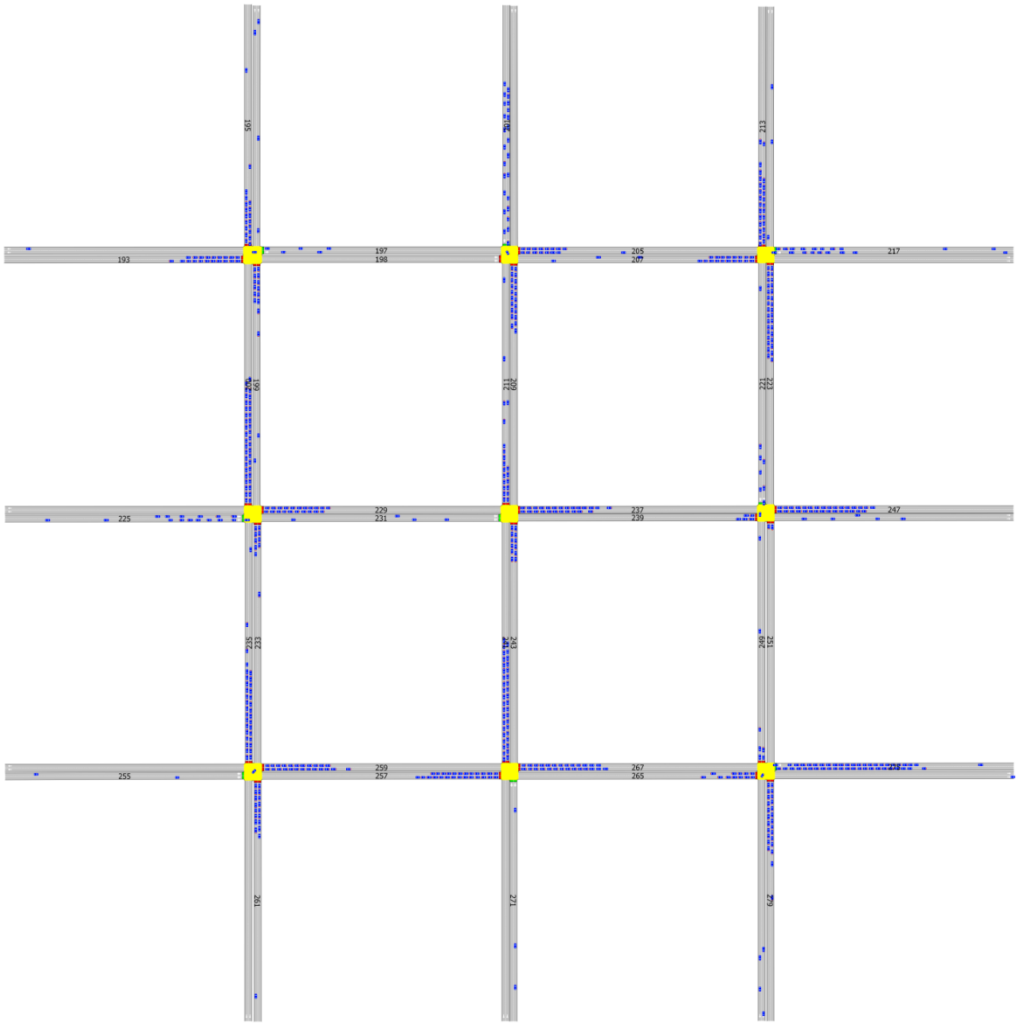

Figure 3. Microscopic traffic simulation

The movement of a vehicle depends on the external properties of the vehicle (vehicle type), such as length, width, maximum speed, and acceleration, as well as internal characteristics of the human driver including reaction time (sec) and reaction time at stop (sec) (Casas et al. 2010). The parameters incorporated in the vehicle movements are shown in table 2.

Table 2. Parameters of vehicles

Properties	Mean value	Standard deviation
Reaction time	1 sec	0.0 sec
Reaction time at stop	1.35 sec	0.0 sec
Length	4 m	0.5 m
Width	2 m	0.0 m
Maximum speed	100 km/h	10 km/h
Maximum acceleration	3 m/s^2	0.2 m/s^2
Maximum deceleration	6 m/s^2	0.5 m/s^2

The driving speed (v_d) is determined by taking four factors into account: maximum speed (v_m), section speed limit (v_s), speed acceptance factor (f_s), and speed of the following vehicle. The section speed limit is the maximum allowed speed of the vehicles passing a section. The maximum allowed speed of all the sections is 50 km/h. The speed acceptance factor shows the degree of accepting the speed limits of sections by a driver. The value of f_s for each vehicle is drawn from a Gaussian distribution function with the mean of 1.1 and standard deviation of 0.1. The driving speed is calculated by $v_d = minimum(v_m, f_s \times v_s)$. Also, the driving speed momentary changes based on the speed of the vehicle ahead. If the vehicle in front has a lower speed, the follower should reduce its driving speed according to the car following model (Gipps 1981) or change its driving lane (Gipps 1986).

The traffic simulation is carried out for 300 hours. Also, each one hour is referred to as an episode. Since all the intersections are 4-way crossroads, the system contains homogeneous CRL-TSCs. A signals group is assigned to each approaching street as shown in figure 4. Thus, each CRL-TSC controls a signalized intersection which has four phases each assigned to an approaching link.

124

Figure 4. Order of the phases

The point that is important in the learning of CRL-TSCs is the value of the learning rate (α) and discount factor (γ). The best found values for the learning rate for tile coding, RBF, and TSF are 0.1, 0.075, and 0.075 respectively. Also, the discount factor is set to 0.99. It should be noted that these values were obtained by trial and error. Moreover, the value of ω (Boltzmann parameter) increases from 0.0 to 10.0 during the first 200 episodes (training period) and then it is kept constant at 10.0 over the last 100 episodes (test period) in order to evaluate the learning performance of the system.

Another point that is the key to success of the CRL-TSCs' effectiveness is the number of tiles, RBFs, and TSFs. Choosing the wrong number of tiles, RBFs, and TSFs can ruin the generalization property of CRL-TSCs. Hence, in order to show the impact of the set-up of tiles, RBFs, and TSFs on the learning performance and find the optimal ones, the performance of CRL-TSCs are evaluated based on different numbers of tiles (3, 5, 7, and 9), RBFs (3, 5, 7, and 9), and TSFs (3, 5, 7, and 9) on each dimension of the state space. Due to the space limitations in this paper, the results of investigating the optimal number of tiling layers in the tile coding approach are not presented and it is directly set to 3. In fact, 3 tilings worked best in preliminary experiments.

Since each CRL-TSC controls one intersection and each intersection has four approaching streets, the state space has a dimension of 5 (index of the current green phase as well as 4 approaching streets). The total number of features for each CRL-TSC based on the RBFs and TSFs approaches is $n = m \times m \times m \times m \times ph$, where m is the number of features on each dimension (e.g. 3, 5, 7, and 9) and ph = 4 is the number of phases of the traffic signals. Also, the total number of features based on the tile coding approach is $n = k \times (m \times m \times m \times m \times ph)$, where k = 3 is the number of tiling layers.

The traffic simulation has been implemented using AIMSUN. Three indexes, average travel time (sec/km), average stop time (sec/km), and average stop numbers (#/veh/km) are used to assess the performance of CRL-TSCs. The average travel time is the average time that a vehicle requires to travel one kilometer. The average stop time is the average time that a vehicle stays at a standstill status in traveling one kilometer inside the traffic network. The average stop number is the average number of stops per vehicle per kilometer.

5-2- Results

Figure 5 presents the performance of CRL-TSCs with 3 tilings and different numbers of tiles (3, 5, 7, and 9) in terms of average travel time. The learning curves are the average of repeated (five times) simulations. As it is clear, 3 tiles has the poorest performance due to the lack of segmentation in the state variables. In other words, the number of features is not high enough in order to provide the CRL-TSCs with fairly flexible generalization ability to adapt to different traffic states. Increasing the number of tiles from 3 to 5 results in considerable improvement in comparison to 5 to 9 tiles. Indeed, the considerable difference between the learning performance of 3 and 5 tiles indicates the pivotal role of the number of tiles and their set-up. It is evident that the average performance of 5, 7, and 9 tiles over the last 100 episodes are very close to each other.

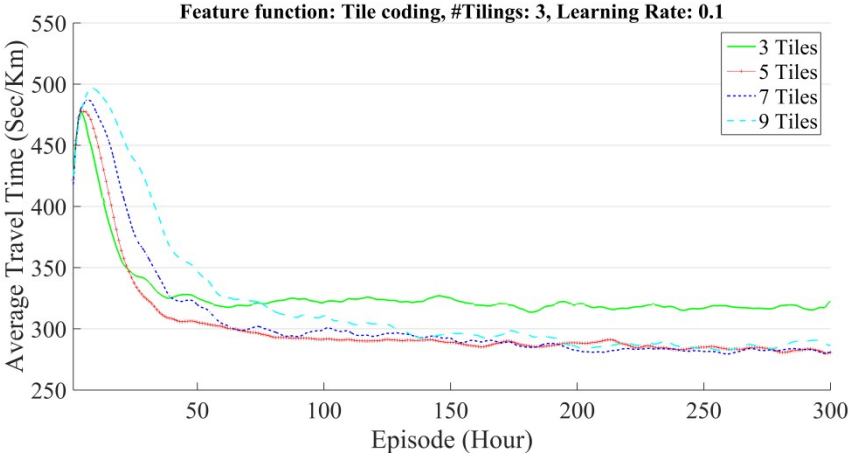

Figure 5. The learning performance of CRL-TSCs for different numbers of tiles for tile coding

The learning curves of CRL-TSCs for different RBFs are depicted in figure 6. 3 RBFs is outperformed by others (5, 7, and 9 RBFs) owing to the insufficient number of features. Similar to tile coding, there is a significant difference between 3 RBFs and others, that proves the importance of the number of RBFs. Also, the performance of 7 RBFs is almost in line with 9 RBFs over the last 100 episodes. Comparing figure 6 with figure 5 reveals that tile coding slightly outperforms RBF in terms of average travel time.

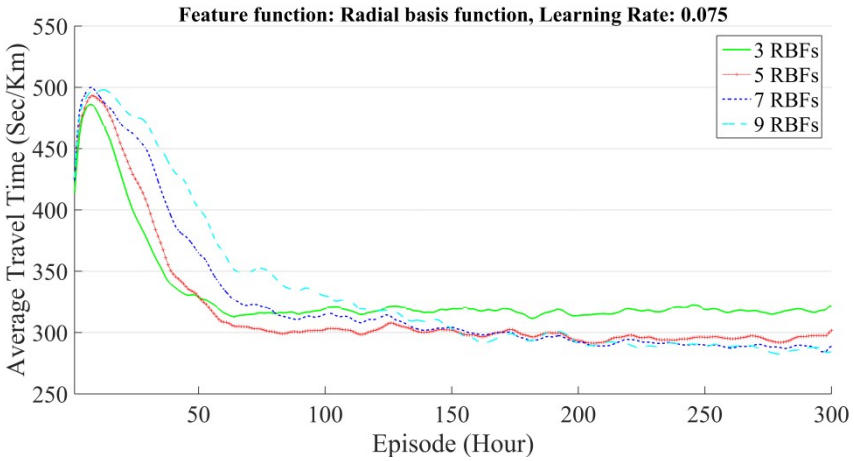

Figure 6. The learning performance of CRL-TSCs for different numbers of RBFs

Figure 7 shows the performance of CRL-TSCs for different numbers of TSFs. Similar to tile coding and RBF, the convergence speed decreases as the number of TSFs increases. This is due to the increase in a number of scalar weights parameters. Comparing figure 5-7 indicates that 3 TSFs has better performance in comparison to 3 RBFs and 3 Tiles.

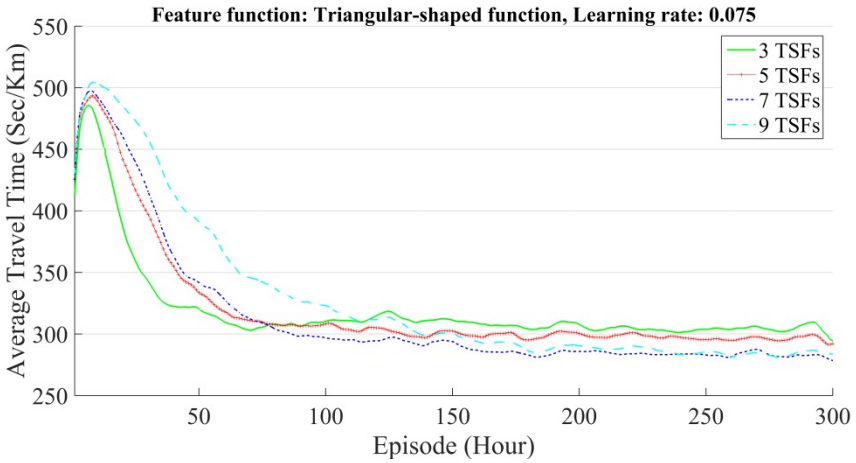

Figure 7. The learning performance of CRL-TSCs for different numbers of TSFs

Table 3 compares the average performance, as measured by the three performance indicators described earlier, of different CRL-TSCs over the last 100 episodes. The best feature functions are shown in boldface. It is evident that 7 features in all three feature functions leads to the best outcome. Also, increasing the number of features from 3 to 7 has improved the performance, but increasing from 7 to 9 could not make the results better. Therefore, increasing the number of tiles does not necessarily lead to an improvement in the performance of the system. Comparing the results reveals that the CRL-TSC with 3 tilings and 7 tiles is the best controller.

Table 3. Average performance of CRL-TSCs for different numbers of tiles, RBFs, and TSFs over the last 100 episodes

Feature Function	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Stop Numbers (#/Veh/Km)
3 Tiles	318±5	231±5	4.52±0.05
5 Tiles	285±4	199±4	4.07±0.05
7 Tiles	**282±4**	**196±4**	**4.02±0.05**
9 Tiles	286±5	200±5	4.02±0.05
3 RBFs	318±6	231±6	4.40±0.07
5 RBFs	295±4	209±4	4.17±0.05
7 RBFs	**289±5**	**203±5**	**4.08±0.06**
9 RBFs	289±6	203±6	4.08±0.06
3 TSFs	304±5	218±5	4.22±0.05
5 TSFs	297±4	211±4	4.21±0.05

7 TSFs	**284±4**	**198±4**	**4.04±0.05**
9 TSFs	286±5	200±5	4.05±0.05

The next issue is the impact of the action space on the performance of CRL-TSCs. The small action space leads to a high learning speed, but at the same time might result in lower flexibility. In fact, it should be determined if increasing the flexibility in the action space results in a significant improvement in CRL-TSC performance. With this end in view, the performance of CRL-TSC with a higher number of actions is evaluated. The new action space has 16 actions with steps of 5 seconds, i.e. [20, 25, 30, …, 90] sec. This allows us to determine whether the shorter green times are crucial in traffic signal control. Figure 8 indicates the performance of CRL-TSC with the new action space and compares it with the initial controller. As it is clear, increasing the number of actions from 8 to 16 does not significantly affect the final performance of CRL-TSCs. Therefore, using the new action space is unreasonable as it just increases the state-action space size.

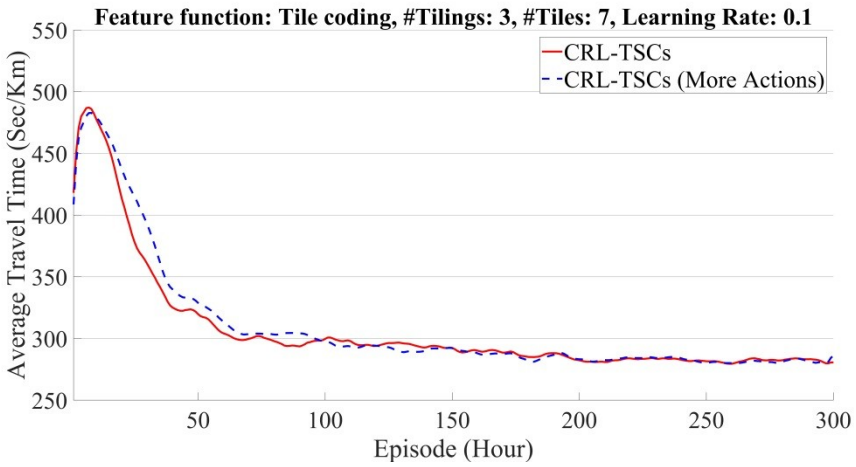

Figure 8. The learning performance of CRL-TSC for two different action spaces

Another issue is the effect of augmenting the state space with the departing links connected to the junction. Considering departing links in the state space provides the CRL-TSCs with the ability to handle the spillback phenomenon and indirect cooperation with the neighboring traffic

signals. On the other hand, it leads to a substantial increase in the state space size. The new state space of each agent has a dimension of 5+D. The first five elements are the index of the current green time and the number of vehicles waiting on each of the four approaching streets. The remaining components indicate the number of vehicles on each departing street. Figure 9 shows the performance of CRL-TSC with the augmented state space and compares it with the initial one. It is evident that considering the departing streets improves the performance of CRL-TSCs.

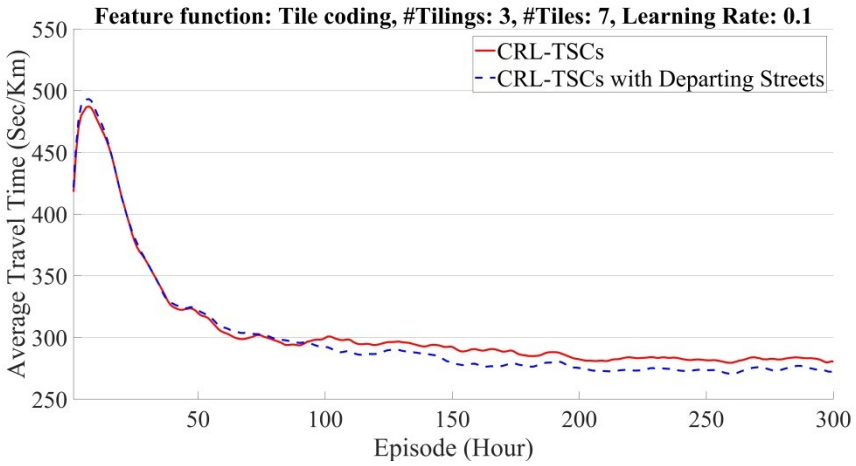

Figure 9. The learning performance of CRL-TSC for two different state spaces

The average performance of CRL-TSCs with the new state space and action space over the last 100 episodes is indicated in table 4. Increasing the number of actions could not significantly affect the learning performance, but adding the departing streets could improve the performance by 3.2%.

Table 4. Comparison of different CRL-TSCs with different state and action spaces

Controller	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Stop Numbers (#/Veh/Km)
1- CRL-TSC	282±4	196±4	4.02±0.04
2- CRL-TSC with More Actions	282±4	196±4	3.97±0.05
3- CRL-TSC with Departing Links	273±4	191±4	3.89±0.05
% Improvement controller 1 vs. 2	0	0	1.2
% Improvement controller 1 vs. 3	3.1	2.6	3.2

6-Discussion

Discrete reinforcement learning methods have been widely used for adaptive traffic signal control (Abdoos et al. 2011, Abdoos et al. 2013). In order to validate the performance of the best CRL-TSCs, we benchmark its performance against standard Q-learning (Watkins and Dayan 1992) and actor-critic (Konda and Borkar 1999). Q-learning is categorized as an off-policy method that learns the values of state-actions on the basis of the optimal policy independent of the policy being followed. During the learning process, the agent stores a particular Q-value for each state-action pair. Equation 12 shows how state-action values are updated (Sutton and Barto 1998).

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha\left[r_{t+1}+\gamma \max_{a'} Q(s_{t+1},a')-Q(s_t,a_t)\right] \quad (12)$$

Where $0 < \alpha \leq 1$ is the learning rate, γ is the discount factor, and s_t is the discretized state. The optimal values of the discount factor and learning rate are 0.99 and 0.01. Unlike Q-leaning, actor-critic is an on-policy method for which state-action values are updated based on the policy being followed. It has a separate memory structure for estimating state-values (the critic) and a mapping from states to a preference value for each action (the actor) (van Otterlo and Wiering 2012). The values of states are updated according to the following equation:

$$V(s_t) \leftarrow V(s_t)+\alpha\left[r_{t+1}+\gamma V(s_{t+1})-V(s_t)\right] \quad (13)$$

In the actor, the values of all state-action pairs are updated by equation 14.

$$P(s_t,a_t) \leftarrow P(s_t,a_t)+\beta\left[r_{t+1}+\gamma V(s_{t+1})-V(s_t)\right] \quad (14)$$

Where $0 < \beta \leq 1$ is the learning rate of the actor and $P(s_t,a_t)$ indicates the tendency to select action a_t in state s_t. The learning rates and discount factor of the actor-critic method are respectively set to 0.01 and 0.99.

For employing discrete reinforcement learning, it is necessary to discretize the state space. In order to do so, each state variable is first discretized into a finite set of regions. Then, the union of all state variables covers the whole state space. In this way, the total number of states is the product of the number of regions for each discretized state variable. In this research, the first four state variables (approaching streets) are discretized into 6 regions and the last state variable (index of the current green phase) is discretized into 4 regions. Thus, the total number of states is $n = 6 \times 6 \times 6 \times 6 \times 4 = 5184$.

The Boltzmann method is used for balancing between exploration and exploitation. Similar to CRL-TSC the Boltzmann parameter gradually increases from 0.0 to 10.0 over the first 200 episodes and then it is kept constant at 10.0 over the last 100 episodes. Figure 10 compares the performance of the best CRL-TSC with standard Q-learning and actor-critic in terms of average travel time.

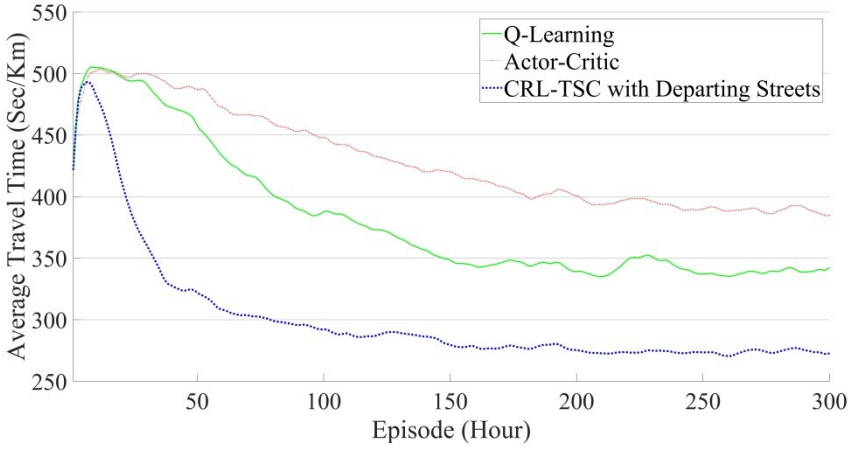

Figure 10. Comparing the performance of the best CRL-TSC with standard Q-learning and actor-critic

As depicted in figure 10, the best CRL-TSC outperforms the other reinforcement learning methods. Table 5 compares the average performance of these controllers based on average travel time, stop time, and stop numbers over the last 100 episodes. It is clear that the best CRL-TSC saves average travel time by 30% and 20% in comparison to actor-critic and Q-learning.

Increasing the air pollution and fuel consumption are two destructive consequences of inefficient traffic control systems. In this paper, we evaluate the impact of the best CRL-TSC on decreasing air pollution and fuel consumption. With this end in mind, fuel consumption and emission rates (HC, CO, and NO_x) are incorporated into the microscopic traffic simulation.

Vehicle specific power (VSP), a technique featuring engine power, is utilized to model the vehicle fuel consumption rate. VSP shows how the vehicle operating conditions affect the fuel consumptions. VSP for typical light-duty vehicles depends on the speed, acceleration (deceleration) and roadway slope on the basis of second-by-second cycles (Jiménez-Palacios 1999). According to the study conducted by the Air Quality Control Company of Tehran (AQCC 2014), the exponential function fits the relationship well between the VSP and the fuel consumption rate (Equation 15).

$$FC\left(\frac{lit}{Sec}\right) = a \times e^{b \times VSP} + c \times VSP + d \quad (15)$$

In equation 15; a, b, c, and d are the constant coefficients. The constant coefficients have been calibrated by the Air Quality Control Company of Tehran (AQCC 2014).

Emission rates of vehicles depend on different parameters such as vehicle speed, vehicle mileage, engine temperature and vehicle load. However, only the first parameter (i.e. vehicle speed) is considered for modeling emission rates in the present research. A 6th order polynomial function is employed for modeling the emission rate (Equation 16) (Boulter et al. 2009, TRL 1999).

$$Emission\,Rate\left(\frac{gr}{km}\right) = \frac{A + Bv + C v^2 + D v^3 + E v^4 + F v^5 + G v^6}{v} \quad (16)$$

Where A - G are coefficients and v is the speed of the vehicle in km/h. The coefficients in equation 16 have been calibrated for different air pollutants (CO, HC, and NOx) by AQCC (2014). The total fuel consumption and traffic-generated air pollution for the best CRL-TSC, actor-critic, and Q-learning are presented in figure 11. It is clear that the CRL-TSC has the best performance with regard to air pollution and fuel consumption.

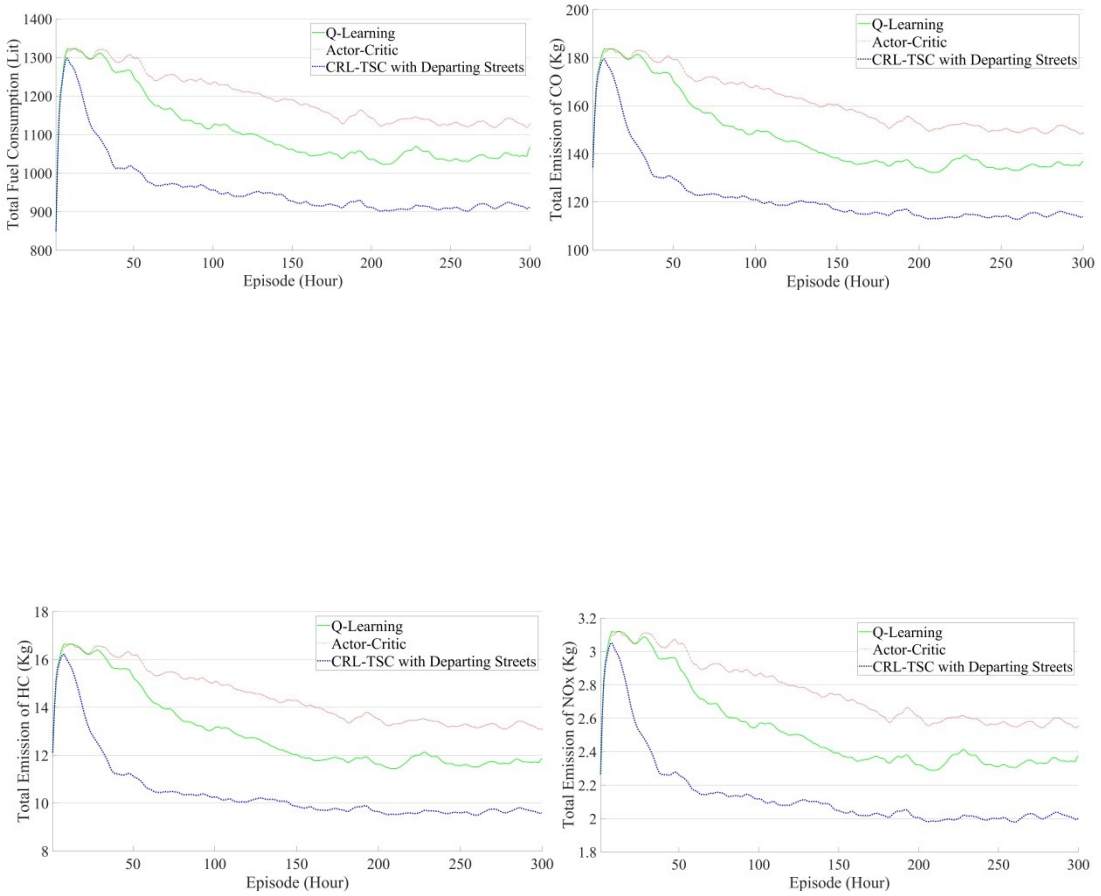

Figure 11. Learning curves of the best CRL-TSC, discrete state Q-learning, and actor-critic in terms of fuel consumption and emissions of air pollutants

Table 5. Comparison of the best CRL-TSC, standard Q-learning and actor-critic

Controller	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Stop Numbers (#/Veh/Km)
Actor-critic	392±7	304±7	4.44±0.06
Q-learning	340±8	254±7	4.26±0.07
CRL-TSC with Departing Links	273±4	191±4	3.89±0.05

% Improvements CRL-TSC vs. actor-critic	30.4	37.2	12.39
% Improvements CRL-TSC vs. Q-learning	19.70	24.8	8.7

Also, table 6 compares the average performance of these controllers in terms of fuel consumption and air pollution over the last 100 episodes. It is clear that CRL-TSC decreases 20% total fuel consumption and 22% total emission of NO_x relative to actor-critic as well as 13% fuel consumption and 14% emission of NO_x in comparison with Q-learning.

Table 6. Comparison between the best CRL-TSC with Q-learning and actor-critic

Controller	Total Fuel (Lit)	Total CO (Kg)	Total HC (Kg)	Total NO_x (Kg)
Actor-Critic	1131±22	150±3.1	13.3±0.29	2.58±0.05
Q-Learning	1042±24	135±3.3	11.7±0.31	2.34±0.06
CRL-TSC with Departing Links	910±18	114±2.4	9.6±0.21	2.00±0.04
Improvements CRL-TSC vs. actor-critic %	19.5	24	27.8	22.5
% Improvements CRL-TSC vs. Q-learning	12.7	15.5	17.9	14.5

So far CRL-TSCs do not consider the spatial distribution of vehicles along the approaching streets. In order to provide them with this ability, each street is split into two parts and the number of vehicles per street-segment is used as state variables. In fact, the vehicles that are more than 100 meters away from the intersections are considered in separate variables from the closer ones. The (near)-optimal number of tilings and tiles are found to be 3. Figure 12 shows the learning performance of the new CRL-TSC and compares it with the former design. It is clear that considering the spatial distribution can slightly improve the performance, although the improvement is not significant.

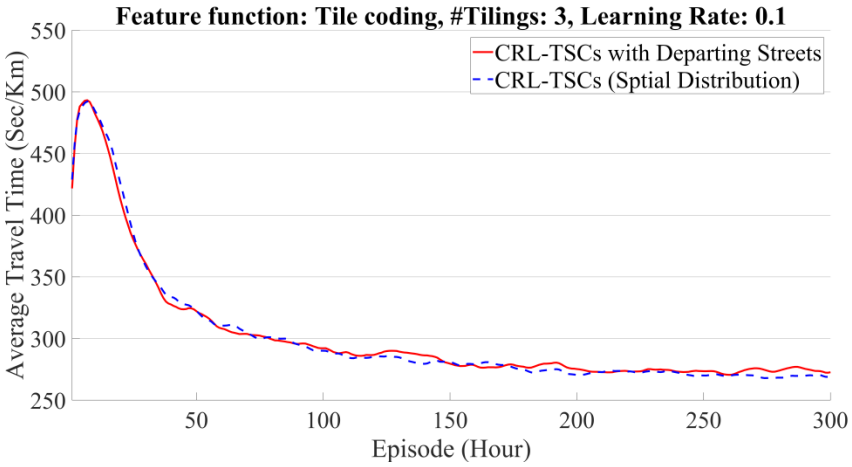

Figure 12. The learning performance of CRL-TSC for two different state spaces

The best CRL-TSC is benchmarked against an optimized fixed-time controller in order to verify the performance of the proposed method. In the fixed time controller, the timing of signals is always fixed no matter how the traffic loads change. As shown in table 7, the best CRL-TSC results in a lower average travel time, stop time, and fuel consumption. The most notable improvement is average stop time.

Table 7. Comparison of the best CRL-TSC and an optimized fixed-time controller

Controller	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Fuel (Lit)
Best CRL-TSC	273	191	910
Fixed-time	323	237	968
% Improvement best CRL-TSC vs. fixed-time	15	19	6

7-Conclusion

This paper described a continuous state reinforcement learning traffic signal controller (CRL-TSC). CRL-TSC has the ability of generalization which enables it to perform accurately on unseen states. The tile coding, RBFs, and TSFs approaches as different function approximation

methods were applied in CRL-TSC to tackle the challenges arising from the continuous state space in the traffic network.

A small traffic network (3×3 grid network) was employed for the thorough evaluation of CRL-TSC and the influence of the parameters. CRL-TSCs were designed in such a way that they can be deployed by basic existing traffic infrastructures in developing countries such as Iran. Owing to the distributed feature of CRL-TSCs, they are extendable to a traffic network with many intersections. Results show that function approximation is crucial in the performance of CRL-TSCs. The excessive rise and reduction in the number of features may ruin the generalization property of CRL-TSCs. Also, the results indicate that the best CRL-TSC is feasible and effective, compared with standard Q-learning and actor-critic and it greatly reduces the average travel time for vehicles as well as fuel consumption and emissions.

Reference

Abdoos, M., Mozayani, N., and Bazzan, A.L.C. 2011. Traffic Light Control in Non-stationary Environments based on MultiAgent Q-learning. *In* 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). pp. 1580-1585.

Abdoos, M., Mozayani, N., and Bazzan, A.L.C. 2013. Holonic multi-agent system for traffic signals control. Engineering Applications of Artificial Intelligence **26**(5–6): 1575-1587.

Abdoos, M., Mozayani, N., and Bazzan, A.L.C. 2014. Hierarchical control of traffic signals using Q-learning with tile coding. Applied Intelligence **40**(2): 201-213.

Albus, J.S. 1975. A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller (CMAC). Journal of Dynamic Systems, Measurement, and Control **97**: 220-227.

AQCC. 2014. The coefficient of emissions in the warm state for gasoline light duty vehicles of Iran Air Quality Control Company of Tehran Municipality Tehran.

Aslani, M., Mesgari, M.S., and Wiering, M. 2017. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. Transportation Research Part C: Emerging Technologies **85**: 732-752.

Baird, L. 1999. Reinforcement Learning Through Gradient Descent, Computer Science, Carnegie Mellon University Pittsburgh.

Bhatta, B. 2010. Analysis of Urban Growth and Sprawl from Remote Sensing Data. Springer, Verlag Berlin Heidelberg.

Bi, Y., Srinivasan, D., Lu, X., Sun, Z., and Zeng, W. 2014. Type-2 fuzzy multi-intersection traffic signal control with differential evolution optimization. Expert Systems with Applications **41**(16): 7338-7349.

Bishop, C.M. 1995. Neural networks for pattern recognition. Clarendon press, Oxford.

Boulter, P., Barlow, T., and MacCrae, I. 2009. Emission Factors 2009: Report 3 – Exhaust Emission factors for Road Vehicles in the United Kingdom. TRL, United Kingdom.

Casas, J., Ferrer, J.L., Garcia, D., Perarnau, J., and Torday, A. 2010. Traffic Simulation with Aimsun. *In* Fundamentals of Traffic Simulation. *Edited by* J. Barceló. Springer New York, New York, NY. pp. 173-232.

Chiou, Y.-C., and Huang, Y.-F. 2013. Stepwise genetic fuzzy logic signal control under mixed traffic conditions. Journal of Advanced Transportation **47**(1): 43-60.

Chowdhury, M.A., and Sadek, A.W. 2003. Fundamentals of Intelligent Transportation Systems Planning Artech House, Norwood, MA.

El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. 2013. Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and

Large-Scale Application on Downtown Toronto. IEEE Transactions on Intelligent Transportation Systems **14**(3): 1140-1150.

Gartner, N.H. 1983. OPAC: A demand-responsive strategy for traffic signal control. Transportation Research Record: Journal of the Transportation Research Board **906**: 75–81.

Gipps, P.G. 1981. A behavioural car-following model for computer simulation. Transportation Research Part B: Methodological **15**(2): 105–111.

Gipps, P.G. 1986. A model for the structure of lane-changing decisions. Transportation Research Part B: Methodological **20**(5): 403–414.

Head, K.L., Mirchandani, P.B., and Sheppard, D. 1992. Hierarchical framework for real-time traffic control. Transportation Research Record **1360**: 82–88.

Henry, J.J., Farges, J.L., and Tufal, J. 1983. The PRODYN real-time traffic algorithm. IFAC Proceedings Volumes **16**(4): 305–310.

Hunt, P.B., Robertson, D.I., Bretherton, R.D., and Winton, R.I. 1981. SCOOT - a traffic responsive method of coordinating signals, Crowthorne, U.K.

Jiménez-Palacios, J.L. 1999. Understanding and quantifying motor vehicle emissions with vehicle specific power and tunable infrared laser differential absorption spectrometer remote sensing, Mechanical Engineering, Massachusetts Institute of Technology, Massachusetts.

Kaelbling, L.P., Littman, M.L., and Moore, A.W. 1996. Reinforcement Learning: A Survey. Journal of Artifcial Intelligence Research **4**: 237-285.

Konda, V.R., and Borkar, V.S. 1999. Actor-critic like learning algorithms for Markov decision processes. SIAM Journal on Control and Optimization **38**(1): 94-123.

Mamdani, E.H. 1974. Application of fuzzy algorithms for control of simple dynamic plant. Electrical Engineers, Proceedings of the Institution of **121**(12): 1585-1588.

Medina, J.C., and Benekohal, R.F. 2012. Q-learning and Approximate Dynamic Programming for Traffic Control: A Case Study for an Oversaturated Network. *In* 91st Annual Meeting of the Transportation Research Board, Washington, D.C.

Medina, J.C., Hajbabaie, A., and Benekohal, R.F. 2010. Arterial traffic control using reinforcement learning agents and information from adjacent intersections in the state and reward structure. *In* 13th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, Funchal. pp. 525 - 530.

Oliveira, L.B.d., and Camponogara, E. 2010. Multi-agent model predictive control of signaling split in urban traffic networks. Transportation Research Part C: Emerging Technologies **18**(1): 120–139.

Prashanth, L., and Bhatnagar, S. 2011. Reinforcement Learning With Function Approximation for Traffic Signal Control. IEEE Transactions on Intelligent Transportation Systems **12**(2): 412-421.

Richter, S. 2006. Learning Road Traffic Control: Towards Practical Traffic Control Using Policy Gradients, Institute of informatcis, Albert Ludwig University of Freiburg, Germany.

Rodrigue, J.-P., Comtois, C., and Slack, B. 2017. The Geography of transport system. Routledge, New York.

Salkham, A., Cunningham, R., Garg, A., and Cahill, V. 2008. A Collaborative Reinforcement Learning Approach to Urban Traffic Control Optimization, 9-12 Dec. 2008, IEEE, pp. 560-566.

Samarasinghe, S. 2016. Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition. Auerbach Publications, New York.

Schwartz, H.M. 2014. Multi-Agent Machine Learning: A Reinforcement Approach. Wiley, New Jersey.

Sims, A.G., and Dobinson, K.W. 1980. The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits. IEEE Transactions on Vehicular Technology **29**(2): 130-137.

Srinivasan, D., Choy, M.C., and Cheu, R.L. 2006. Neural Networks for Real-Time Traffic Signal Control. IEEE Transactions on Intelligent Transportation Systems **7**(3): 261-272.

Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., and Bakker, B. 2005. Reinforcement learning of traffic light controllers adapting to traffic congestion, Brussels, 2005, pp. 216–223.

Sutton, R.S. 1988. Learning to Predict by the Methods of Temporal Differences. Machine Learning **3**(1): 9-44.

Sutton, R.S., and Barto, A.G. 1998. Reinforcement Learning: An Introduction. MIT Press, Cambridge.

Szepesvári, C. 2010. Algorithms for Reinforcement Learning. Morgan & Claypool Publishers.

Takagi, T., and Sugeno, M. 1985. Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics **SMC-15**(1): 116-132.

Thorndike, E.L. 1911. Animal intelligence; experimental studies The Macmillan company, New York.

TRL. 1999. Methodology for calculating transport emissions and energy consumption. Transport Research Laboratory, Crow thorne, United Kingdom.

van Otterlo, M., and Wiering, M. 2012. Reinforcement Learning and Markov Decision Processes. *In* Reinforcement Learning: State-of-the-Art. *Edited by* M. Wiering and M. van Otterlo. Springer, Berlin, Heidelberg. pp. 3-42.

Watkins, C., and Dayan, P. 1992. Q-learning. Machine learning **8**(3): 279–292.

Wiering, M. 2000. Multi-agent reinforcement learning for traffic light control, Stanford, CA, pp. 1151-1158.

Zadeh, L.A. 1965. Fuzzy sets. Information and Control **8**(3): 338-353.

Tables

448 Table 1. Traffic network configuration

Properties	Value
number of intersections	9
number of links	48
average length of links	250
number of lanes per links	2
maximum speed	50 km/h
number of input/output centroids	12
arrival distribution	Gaussian
simulation duration	700 hours

449

450

451 Table 2. Parameters of vehicles

Properties	Mean value	Standard deviation
Reaction time	1 sec	0.0 sec
Reaction time at stop	1.35 sec	0.0 sec
Length	4 m	0.5 m
Width	2 m	0.0 m
Maximum speed	100 km/h	10 km/h
Maximum acceleration	3 m/s^2	0.2 m/s^2
Maximum deceleration	6 m/s^2	0.5 m/s^2

452

453

454

455 Table 3. Average performance of CRL-TSCs for different numbers of tiles, RBFs, and TSFs over the last 100

456 episodes

Feature Function	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Stop Numbers (#/Veh/Km)
3 Tiles	318±5	231±5	4.52±0.05
5 Tiles	285±4	199±4	4.07±0.05
7 Tiles	**282±4**	**196±4**	**4.02±0.05**
9 Tiles	286±5	200±5	4.02±0.05
3 RBFs	318±6	231±6	4.40±0.07
5 RBFs	295±4	209±4	4.17±0.05
7 RBFs	**289±5**	**203±5**	**4.08±0.06**
9 RBFs	289±6	203±6	4.08±0.06
3 TSFs	304±5	218±5	4.22±0.05
5 TSFs	297±4	211±4	4.21±0.05
7 TSFs	**284±4**	**198±4**	**4.04±0.05**

9 TSFs	286±5	200±5	4.05±0.05

Table 4. Comparison of different CRL-TSCs with different state and action spaces

Controller	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Stop Numbers (#/Veh/Km)
1- CRL-TSC	282±4	196±4	4.02±0.04
2- CRL-TSC with More Actions	282±4	196±4	3.97±0.05
3- CRL-TSC with Departing Links	273±4	191±4	3.89±0.05
% Improvement controller 1 vs. 2	0	0	1.2
% Improvement controller 1 vs. 3	3.1	2.6	3.2

Table 5. Comparison of the best CRL-TSC, standard Q-learning and actor-critic

Controller	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Stop Numbers (#/Veh/Km)
Actor-critic	392±7	304±7	4.44±0.06
Q-learning	340±8	254±7	4.26±0.07
CRL-TSC with Departing Links	273±4	191±4	3.89±0.05
% Improvements CRL-TSC vs. actor-critic	30.4	37.2	12.39
% Improvements CRL-TSC vs. Q-learning	19.70	24.8	8.7

Table 6. Comparison between the best CRL-TSC with Q-learning and actor-critic

Controller	Total Fuel (Lit)	Total CO (Kg)	Total HC (Kg)	Total NO_x (Kg)
Actor-Critic	1131±22	150±3.1	13.3±0.29	2.58±0.05
Q-Learning	1042±24	135±3.3	11.7±0.31	2.34±0.06
CRL-TSC with Departing Links	910±18	114±2.4	9.6±0.21	2.00±0.04
Improvements CRL-TSC vs. % actor-critic	19.5	24	27.8	22.5
% Improvements CRL-TSC vs. Q-learning	12.7	15.5	17.9	14.5

Table 7. Comparison of the best CRL-TSC and fixed-time controllers

Controller	Avg. Travel Time (Sec/Km)	Avg. Stop Time (Sec/Km)	Avg. Fuel (Lit)
Best CRL-TSC	273	191	910
Fixed-time	323	237	968
% Improvement best CRL-TSC vs. fixed-time	15	19	6

Figure captions